NORD-1

COMPUTER SYSTEMS

SINTRAN II

OPERATOR'S GUIDE

February 1973

A/S NORSK DATA-ELEKTRONIKK

Erich Mogensøns vei 38, Oslo 5

REVISION RECORD

Revision	Notes
8/73	Original Printing

USING THIS MANUAL

+‡+

SINTRAN II Operator's Guide is intended for the persons being in direct contact with the computer running SINTRAN. The manual therefore contains summaries of the use of the subsystems and explanations to all the printouts and error messages that can occur. A description of loading and starting SINTRAN is also including. Chapter 6 shows where to find information in various error situations.

--ooOoo--

TABLE OF CONTENTS

+++

1 INTRODUCTION

1.1 Definition of the SINTRAN Monitor System

The SINTRAN Monitor System is a real time, multi-programming, operating system for the NORD-1 computer. The minimum configuration includes a standard NORD-1 computer with a multi-level interrupt system and 8K of core. The configuration may include additional core, a mass storage device (drum or disk) for programs and data, and peripheral devices.

The main application of the SINTRAN Monitor System is considered to be process control, but it also has other real time applications.

1.2 Terms

RT-programs : Real time programs, clock-, interrupt- or program-controlled programs.

Priority : A number p. $0 \leq p \leq 255$ assigned to each RT-program. An RT-program will interrupt an RT-program with a lower priority number.

RT-description : A data element of nine locations associated with each RT-program. It contains priority, time information and start address of the RT-program.

RT-name : A pointer to the first location of the RT-description. It is used whenever the RT-program is referred to, e.g., as a parameter of a monitor call.

Time queue : A queue of RT-programs scheduled for execution of a future time.

Execution queue : A queue of RT-programs ready for execution, waiting for RT-programs of higher priorites to be finished. An RT-program in the time queue will be transferred to the execution queue when its time has expired.

An RT-program will leave the execution queue when it is started.

2 THE SINTRAN MONITOR SYSTEM AND ITS ENVIRONMENT

2.1 Hardware Configurations

2.1.1 Machine Requirements

The following is the minimum configuration required:

- NORD-1 CPU with multi-level interrupt system (excluding floating point hardware), (or NORD-2B CPU with instruction and multi-level interrupt system simulators).
- Real time clock.
- 8K main memory.
- Teletype.

2.1.2 Additional Hardware used

The following variety of additional hardware may be used:

- Memory protect system, used for background programming or debugging of real time application programs.
- Floating point hardware, necessary if the optional function TIME is included.
- Additional main memory. The standard SINTRAN Monitor System can use a main memory up to 65K. For more memory, a few modifications must be included.
- Mass storage - disk or drum.
- Paper tape readers and punches.
- Additional Teletypes.
- Card readers.
- Line printers.
- Modem controls for low and medium speed communication lines.
- Data links to other computers.
- Process interface equipment; digital I/O, analog/digital and digital/analog converters.

2.2 The Software

The Monitor schedules RT-programs and supervises their execution.

An alphanumeric input/output system buffers and processes characters by interrupt-controlled driver routines for the different devices.

An Operator Communication program accepts commands written on a Teletype, controlling the system.

Manual : SINTRAN II Users' Guide

MACM is a mass storage oriented version of the MAC assembler putting the generated code into a core image on the mass storage. It is used for assembling the SINTRAN Monitor System.

Manual : MACM -
MAC MASS STORAGE ASSEMBLER

The real time loader or RT Loader is able to load programs in binary relocatable format (BRF). The programs may be output from the RT-FORTRAN compiler or the MAC assembler.

New programs may be loaded while the system is on-line.

Manual : SINTRAN II Real Time Loader

A version of the MAC assembler, MACD, may be used on-line for debugging purposes. Breakpoints may be inserted in running RT-programs. Programs may be assembled to core or mass storage.

Manual : MACD -
MAC DEBUGGING ASSEMBLER

An RT-FORTRAN Compiler (FORTRAN II or FORTRAN IV version) produces re-entrant code, which may run as RT-programs under the SINTRAN Monitor.

Manual : 1. FORTRAN II
2. NORD FORTRAN IV Reference Manual

2.3 System Generating

When the monitor system is to be used for a new computer configuration, the system programs should be assembled together with some information about the configuration.

The parameters to be determined follow.

For the monitor:

- The area in core for the monitor.
- The hardware interrupt levels to be used.
- The level for the monitor and the level for the RT-programs.
- The interrupt groups to be enabled.
- The basic time unit.
- The number of peripheral devices.
- The area for debug information in case of stack overflow.
- The number of software blocks per coreload.
- The number of hardware blocks per software block.
- The start of the coreload area in core.
- The start of the area on mass storage occupied by coreloads.
- The number of data files on mass storage.

For each peripheral device:

- The buffer capacity.
- The logical unit number.
- The alternative unit.

Two tables have to be set up:

- An interrupt identification table, with one element for each interrupt line.
- A logical unit table, with one element for each logical unit. with reference to the corresponding periperal device.

A list of the names of the monitor subroutines which are wanted should also be set up.

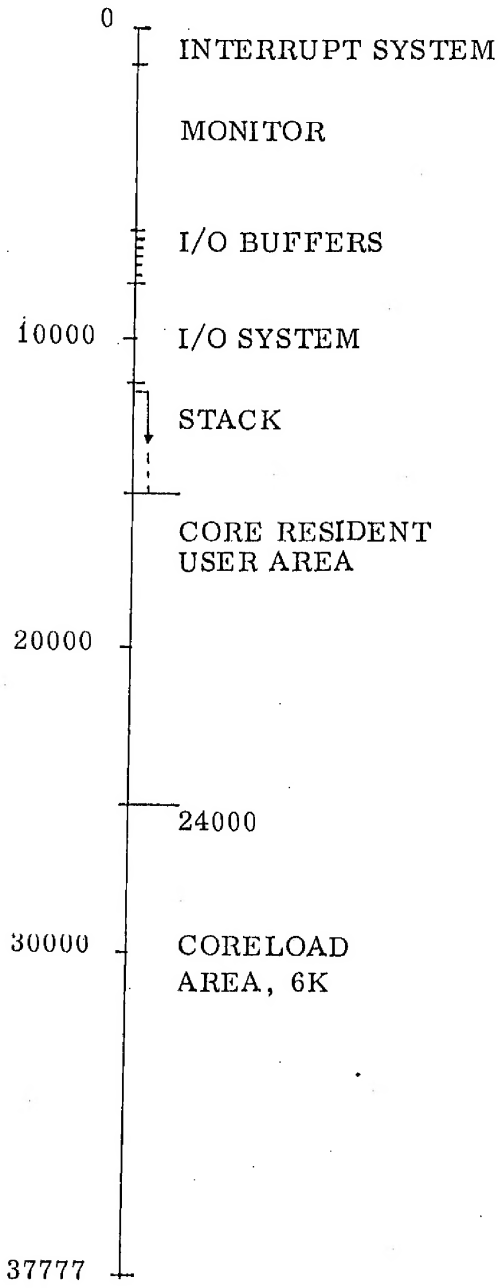
Detailed description of the parameters and system generation procedure is found in the manual:

GENERATING SINTRAN-II.

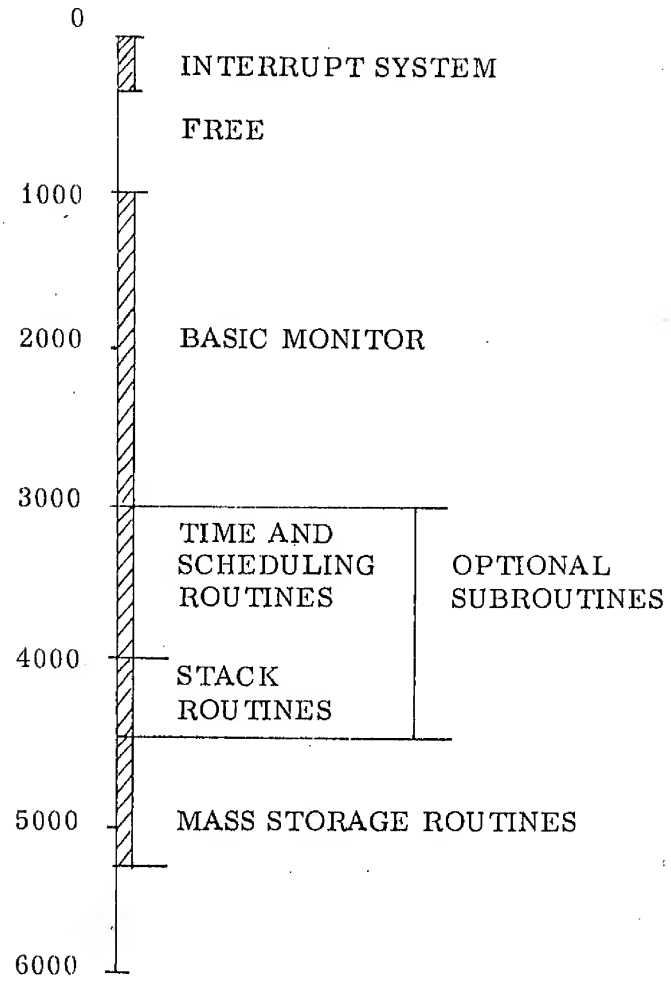
2.4 Allocating Main Memory and Mass Storage

Here is an example of main memory and mass storage allocation with 16K main memory and 256K drum.

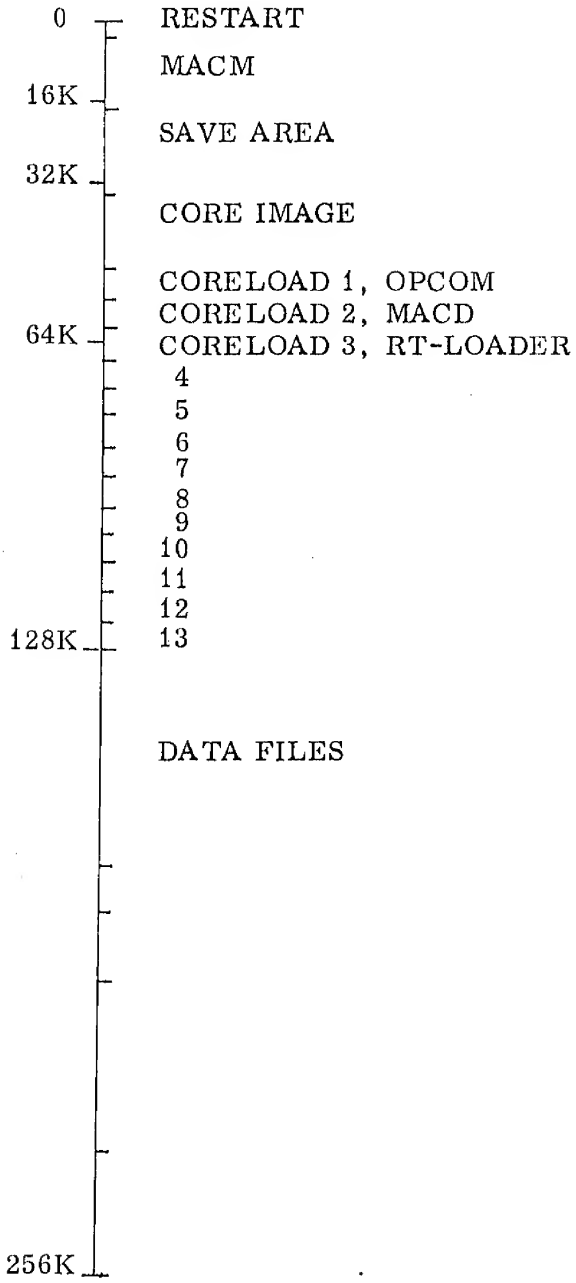
MAIN MEMORY LAYOUT



MONITOR MAGNIFIED:



DRUM LAYOUT:


0
RESTART
MACM
16K
SAVE AREA
32K
CORE IMAGE
CORELOAD 1, OPCOM
CORELOAD 2, MACD
64K
CORELOAD 3, RT-LOADER
4
5
6
7
8
9
10
11
12
128K
13
DATA FILES
256K

2.5 Facilities provided by the SINTRAN Monitor System

2.5.1 Scheduling Real Time Programs

The traditional way of scheduling real time programs for execution is on direct signal (interrupt) from the outside word. In SINTRAN, connections between interrupt lines and real time programs may be made dynamically. However, scheduling also by time has proved to be very much used. Therefore, SINTRAN is equipped with facilities for scheduling programs at a particular time of day or during a time interval (delayed start). Repeated execution may also be specified.

Programs may be placed in permanent core or on a mass storage. The monitor will bring a mass storage program into core when it is to be started.

To allow a wide range of execution times, priorites are assigned to the programs, so that a program of high priority can interrupt a program of low priority.

Mechanisms for re-entrancy are included.

2.5.2 Input/Output

Alphanumeric input/output is interrupt-controlled, so that the peripheral devices may operate independently. Peripheral devices are specified by logical unit number.

The input/output system, which is buffered, is designed to handle even communication lines.

2.5.3 Processing Aids

The RT-FORTRAN compiler processes real time FORTRAN programs, giving re-entrant subroutines.

The MAC assembler accepts symbolic programs and subprograms.

The Real Time Loader can load new program units or replace old units while the system is running.

2.5.4 Check-out Aids

The Operator Communication program allows the operator to examine and alter locations, start or stop programs, etc.

The MACD on-line assembler permits insertion of instructions and breakpoints in running programs.

Undebugged programs may run in memory protect mode so as not to disturb the rest of the system.

2.5.5 Installation Management Aids

Several system parameters may be measured; for instance, the worst case start-up delay for a chosen program.

3 THE OPERATOR'S INTERFACE WITH THE SYSTEM

The following is a summary of the tasks of the operator during the phases of the system.

a) Loading the System

The SINTRAN system generated for the spesific configuration comes as a rebootable paper tape, which can be loaded in an empty NORD-1 computer. If a mass storage is included, the system will be placed in its proper locations on the mass storage. For core only systems, the system will be placed in core, ready for start.

b) Starting

The MACM assembler will be used to bring the core resident part of SINTRAN into core and start if it is in the START location. A core only system will be started by starting the computer in the proper location.

c) Run Time Operations

User programs can be loaded, started and supervised. Undebugged programs can be run under memory protect. Debugging facilities, for example breakpoint and dumps may be used.

d) Error Handling

Several types of errors may show up, more or less serious ones.

4 STANDARD OPERATING PROCEDURES

4.1 Loading the System

4.1.1 Mass Storage Systems

The paper tape has the following form:

)LIST part Blank tape)BPUND parts

The)LIST part contains symbol definitions and should be skipped when loading.

Procedure:

1. Set the hardware device number of the mass storage into location 0.

2. Put the tape in the reader after the)LIST part, and start the reader.

3. Push STOP and LOAD to read the rebootable)BPUND tape.

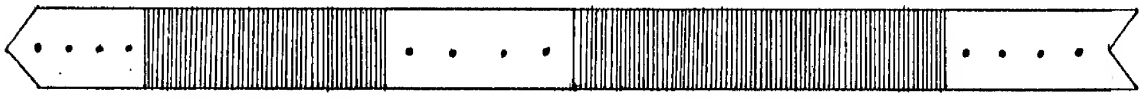
4. When the tape stops, check that (IR) = WAIT 7.
   WAIT 77 means checksum error, WAIT 376 means transfer error. Then T = 0 means "read", and D holds the block address.

5. Repeat step 3 and 4 until all of the tape has been read.

Now the SINTRAN II System is placed on its proper locations on the mass storage. The core resident part is placed on a core image (the working area for MACM).

4.1.2 Core System

The tape has the following form:



)LIST part Blank tape)BPUN part

Procedure:

1. Put the tape in the reader after the)LIST part, and start the reader.

2. Push STOP and LOAD to read the)BPUN part.

3. When the tape stops at its end, check that (IR) = WAIT 0. WAIT 77 means checksum error.

Now the SINTRAN II core system is placed in core, ready for start.

4.2 Start, Stop and Restart

4.2.1 Mass Storage Systems

Initial starting procedure:

1. Get MACM into core.

2. Assemble the)LIST part of the system tape.

3. Write SINTRAN!

4. The core image will then be brought into core, and SINTRAN will be started in the location labeled SINTRAN!

5. SINTRAN now asks for lower and upper stack bounds, which should be answered with the proper octal numbers, terminated by a carriage return.

6. SINTRAN then asks for current time, which should be answered with five decimal numbers, giving minute, hour, day, month and year. The numbers should be delimited by spaces, the last number terminated by a carriage return.

If legal numbers have been given, the interrupt system will be turned on. Thereafter an RT program will be started, writing a message of the form

SINTRAN II <monitor version>/<I/O version>/<version> RUNNING

Example:

This starting procedure may be repeated later, since the core image on the mass storage will normally not be modified while SINTRAN is running. However, the core loads may change, but initial versions can be saved by employing the MACM command)CMOVE.

```
%
% STARTING THE SYSTEM
%

)CLOAD 0
UNDEF SYMBOLS

SINTRAN:014102

SINTRAN!
LOWER STACK BOUND? 11000
UPPER STACK BOUND? 12000
PRESENT TIME? 0 15 28 11 1972
SINTRAN II  721102/ 720925/ ND RUNNING

@* MAC

MAC READY:
!
EXIT MAC!
@* LOAD

PRELIMINARY RT-LOAD IV 7205
FREE AREA ABOVE STACK? 0
NO OF RT-NAMES? 10
CORELOAD? 3

*EX


@* EDIT

HIGH SPEED INPUT?
YYYYYY
*?
*

EXIT EDIT!
@* DATCL
  00029  00001  00015  00028  00011  01972
@* STOP
```

Controlled stop and restart:

The system may be stopped by writing the operator command STOP. The system will then save the registers, turn off the interrupt system and stop by executing the instructions IOF; WAIT.

When CONTINUE is pushed (starting in the location labeled RESTA), the registers will be loaded by their old contents, and the system will proceed where it left.

If the computer is stopped by pushing the button STOP, the system can not be restarted in the location RESTA. An initial start must then be performed.

The stop restart feature is useful when all the user programs have been loaded. Then the following procedure can be used:

1. Stop SINTRAN using the command STOP.

2. Save the core contents to the core image and get MACM into core by using the CTOM tape (see MACM).

3. Punch the necessary mass storage areas by using the)BPUND command in MACM.

4. Restart the system by writing RESTA!

5. Correct the time and day setting by using the commands UPDAT or CLADJ.

The)BPUND tape may be used if a serious system breakdown has occurred. The tape may be read and the system restarted in RESTA.

4.2.2 Core Systems

The system is initially started in the location labeled SINTRAN (the octal address is noted on the tape). The further procedure for setting stack limits and current time is the same as for a mass storage system.

When setting stack limits, the stack is permitted to overlap the start section of the system, except for the 38 lowest stack locations, which will be used by the start section.

If an initial start is to be repeated, the system tape must be read beforehand.

Stop and restart may be performed by means of the operator command STOP and restart in the location RESTA.

Note! If the computer is stopped by pushing of the "STOP" button, SINTRAN cannot be restarted in the location RESTA, because the registers have not been saved.

4.3 Using the Operator Communication

4.3.1 Program Description

The operator communication portion of SINTRAN consists of three RT-programs, a main RT-program with high priority, and two auxiliary RT-programs with low priority.

Main RT-program

The main RT program will be scheduled for execution when an @ is typed on the Teletype. It will then initiate a read operation from the Teletype, being started next time when a carriage return has been given. The command will be analyzed and executed.

Auxiliary RT-programs

An RT-program of low priority will be executing the PUNCH, READ and COMP commands. Another RT program will call MACD, the RT loader or the RT FORTRAN compiler.

4.3.2 Operator Commands

The operator commands may be typed on an on-line Teletype. The Operator Communication may be reached from several Teletypes.

A command consists of a command identifier and 0-5 parameters. The command identifier consists of up to five letters or digits. The parameters may be octal or decimal signed integers. In addition, if an RT-program loaded by the RT loader is to be specified, its symbolic name may be used as parameter. Otherwise, the octal address of its RT description has to be used. Then care must be taken to write the correct number.

The different parts of the command are delimited by spaces. The command is terminated by carriage return.

The procedure for giving operator commands is this:

1) Hit the @ key of an operator Teletype.

2) If the operator communication program is ready, an * (asterisk) will be printed. If no asterisk comes, the program is not ready.

3) When the * has arrived, write the command. The command must end with a carriage return.

4) The commands will be analyzed, and executed if there are no errors.

5) A carriage return/line feed is written, telling that the command is executed.

Commands for examining or changing locations

These commands will have octal parameters. In case of a core only system, the parameter <coreload> will be omitted.

E <coreload> <core address>

Examine location.
The value of the location will be written (octal) on the calling Teletype. If the core address is outside the coreload area, it is assumed to be in permanent core.

Example: E 4 33542

D <new value>

Deposit.
The value is deposited into the location just examined. The function is illegal if the preceding operator function from this Teletype was not EXAM.

Example: D -1

DUMP <coreload> <lower limit> <upper limit>

The octal values of the memory block are dumped on the calling Teletype.

Example: DUMP 0 12000 12020

PUNCH <coreload> <lower limit> <upper limit>

The memory block is punched in a binary format.

Example: PUNCH 0 13020 14000

READ <coreload>

A paper tape produced by PUNCH is read and its contents deposited in memory, or in the coreload specified. A tape produced by the MAC command)BPUN will also be accepted.

In case of wrong check sum the error message 27 will be given.

Note: It is possible to punch an area from one coreload and read it to some other coreload.

Example: READ 5

COMP <coreload>

The paper tape produced by PUNCH (or the MAC command)BPUN) is read and compared to the corresponding memory locations, without changing them. Differences will be printed on the Operator Teletype as 3 octal numbers: address, core contents, tape contents.

In case of check sum error, the error message 27 will be given.

Example: COMP 5

Service Commands

The parameters should be specified as (signed) decimal numbers, except for RT programs, which should be octal addresses or symbolic names.

LOGZ <program name>

The accumulated values of the performance logging will be reset. The parameter indicates the RT-program to be logged.

Example: LOGZ RTP1

LOG Results from the performance logging for RT-program RTP1 will be printed on the Teletype as three decimal integers:

1: Maximum stack locations used since last LOGZ (for all programs).

2: Maximum delay time in the execution queue for the RT-program specified by LOGZ.

3: Maximum execution time for the RT-program.

DATCL

The current date and time will be printed on the operator Teletype, from second up to year.

MAC

The MAC on-line assembler (MACD) is started.
MACD is terminated by writing an exclamation mark (!)

Manual : MACD - MAC Debugging Assembler.

FTN The RT-FORTRAN compiler is started.
FORTRAN is terminated by the command H.

Manual : NORD FORTRAN IV Reference Manual.

LOAD The RT-loader is started.
The loader is terminated by the command EX.

Manual : SINTRAN II Real time Loader.

EDIT

The Conversational Editor is started.
The editor is terminated by two times control-C.

Manual : Conversational Editor.

STOP The hardware level 15 will be activated. Then a system program will save the registers and stop the computer. The system may be restarted from the location that follows the "WAIT" instruction in the STOP-routine (i.e., it is equal to the contents of the P-register at STOP-time).

PIN <log. unit>

This command may be used when continuation of an I/O operation on a device which is timed out (because of a missing interrupt) is desired. The parameter to be used was earlier printed out in the run-time error message 37.

Example: PIN 4

PINC <log. unit>

The device buffer is reset before the timed-out device is restarted (see PIN). This command ought to be used if, for instance, a card reader is timed out because of card crash.

Example: PINC 4

TERM <log. unit>

The "end-of-file" character (27_8) is inserted, and the current record is considered to be ready.

Example: TERM 2

SKIP <log. unit>

Put the output unit in skip-if-full-mode. The output statement will then be ignored if the output buffer is full.

Example: SKIP 9

RSKIP <log. unit>

Take the output unit out of skip-if-full-mode. In case of full buffer, the requesting programs will enter a waiting state.

Example: RSKIP 9

FAIL <log. unit>

The device corresponding to the <log. unit> should no longer be used. The alternative device, if any, will then be used for the I/O operations. If no alternative device is defined, the RT-program will be terminated.

Example: FAIL 2

RFAIL <log. unit>

The device corresponding to <log. unit> should no longer use an alternative device.

Example: RFAIL 2

ALT < log. unit 1> < log. unit 2 >

The device corresponding to < log. unit 2 > is inserted as alternative device for < log. unit 1>. If both are two-way devices, the alternative device will be used both for input and output (after use of FAIL).

Example: ALT 1 9

CONT < program name >

This command should be used to continue execution of a program which executed PAUSE statement.

Example: CONT STAT2

RTOFF < program name >

Starting the RT-program will be inhibited.

RTON < program name >

The RT program may be started.

RTP < prog. name > < memory protect setting >

The memory protect register will be set according to the second parameter. Thereafter the program < prog. name > will be put into the execution queue, like the command RT. The lower 2K of core will alwayse be protected. If the second parameter is zero, the memory protect register contents are taken from the Operator switches.

Example: RTP PTEST 127

The lower 8K will be protected.

WSTK

The names of the RT-programs in the stack will be written, starting with the program with the highest priority.

Example: WSTK

PROGA
PROGB
PROG

WEQ

The names of the RT-programs in the execution queue will be printed, starting with the program with the highest priority. A maximum of six program names will be printed.

Example:
```
WEQ
PR1
PR2
```

WTQ The names of the RT-programs in the time queue will be printed, starting with the program with shortest time to execution. A maximum of six program names will be printed.

Example:
```
WTQ
7712
PTX
PTY
```

WRTS <prog.name>

Static information about the RT-program will be printed: start address, coreload number, priority and a possible connected interrupt line.

Example:

WRTS	PROGN	
SADR	035111	Start address (octal)
CLD	100007	Coreload (octal)
PR1	00033	Priority (decimal)

WRTT <prog.name>

Temporary information about the RT-program will be printed: time, interval, priority, state, and stack limits (if active).

Example:
```
WRTT        PROGM
L.EX  00316S
STCK  013016 013034
```

This means that last execution started 316_{10} seconds ago. At the moment it occupies 16_8 stack locations, i.e., the amount used by the monitor.

Examples of possible additional printout:

EQ	The program is in the execution queue.
TQ 316S	The program is in the time queue, with start scheduled in 316S.
OFF	RTOFF is used.
A	The program has been started by ABSET.

The letter in the time part can be B, S, M or H, meaning basic units, seconds, minutes or hours.

Monitor Commands

Most of the monitor subroutines may be executed as operator commands. The parameters should be specified as (signed) decimal integers, except for RT-programs, which may be octal addresses or symbolic names.

RT <prog. name>

Example: RT STAX

SET <prog. name> <time> <time unit>

Example: SET PP2 18 3

ABSET <prog. name> <second> <minute> <hour>

Example: ABSET PXY 0 30 18

INTV <prog. name> <time> <time unit>

Example: INTV SAMPL 2 2

ABORT <prog. name>

Example: ABORT OPTI

CONCT <prog. name> <int. line no.>

Example: CONCT RESP 9

DSCNT <prog. name>

Example: DSCNT RESP

PRIOR <prog. name> <priority>

If the program is executing or in execution queue, the error message ERROR, NOPASS (no priority assigned) will be given.

Example: PRIOR LPM 19

UPDAT <minute> <hour> <day> <month> <year>

Example: UPDAT 14 10 3 3 1971

CLADJ <time> <time unit>

Example CLADJ 10 2

Error Messages

If the syntax of the command is wrong or a parameter has a value outside certain limits, an error message will be given.

ILL.CH	Illegal character
PAR.NO	Wrong number of parameters
NONE	Non-existing function
PRIORI	Priority outside limits
NOPASS	No priority assigned
DEV.NO	Too big interrupt line number
NOEXAM	E (examine) does not precede D (deposit)
RTSYMB	Symbol not allowed in this case
NORTPR	RT-program name not found
SYMPAR	Symbol used in wrong parameter
RTADR?	RT-program not waiting
PARAM	Illegal parameter value
NOSTRT	Program not started, some other program is using the protect feature.

4.4 Using the MACD Assembler

4.4.1 Purpose

MACD is another version of the MAC assembler. This one is intended for use on-line with the SINTRAN Monitor mainly for basic debugging purposes. It is also intended for use in a mass-storage oriented system.

If a core-only version of MAC is desired, this may easily be accomplished with a few modifications to basic MAC, i.e. input/output IOT's are exchanged with the pertient INCH and OUTCH calls.

MACD works either on resident core or on a specific coreload (located on mass storage) depending on the core address (Current Location Counter). If the address is within the coreload area, the coreload is accessed, else (usually) resident core.

Coreload number zero is an exception to this rule. Whenever the user has specified coreload zero, MACD will access the core-resident core image area on mass storage. This area is generally not used when SINTRAN is working. It is only used for initial loading of resident core when the system is started (or restarted).

To summarize:

MACD always works on an area equal in size to core storage.

When a coreload number different from zero has been specified, the area consists of the two subareas:

1. Resident part of core storage (excluding the coreload area).
2. The specified coreload on mass storage.

With coreload zero, the area equals the core image on mass storage.

4.4.2 Activating MACD

MACD may be called for execution from the Operator Communication Program (OPCOM) with the command:

@ * MAC↵

When MACD is ready one of two messages will be printed on the Teletype:

MAC READY:↵

or

B.P. POINTS TO < address >, < coreload number >↵

The last message indicates that a breakpoint has been set at the address indicated. However, whenever MACD is activated the breakpoint is reset and set again when MACD is terminated.

MACD may also be activated by being called as a subroutine from a breakpoint in a user program. This breakpoint must have been set by a previous activation of MACD. Whenever MACD is activated from a breakpoint, it identifies itself by printing a point (.).

MACD is terminated by typing !. If MACD was activated from OPCOM, the message EXIT MAC↵ is typed, else if called from a breakpoint, the action taken is further specified in the chapter "Break Point".

4.4.3 Use

MACD is mostly used as an ordinary MAC assembler. It has the following options included (MACD 701069):

)FIX	Make all symbols in local table permanent.
)CLOAD < number >	Set coreload number; may be examined by typing, C:.....
)LIST	Punch local table.
)PCL < symbol >	Partial clear of local table.
)SYSDF	Set system definition mode, reset by)LINE. Only equalsign definitions needed (represented in undefined table) are taken care of, others are ignored.

The usual breakpoint functions may also be utilized. To start debugging a user program, the normal procedure is:

1. Call MACD from OPCOM by the command MAC.
2. Set actual coreload number.
3. Read in symbol definitions for program to be debugged.
4. Set a breakpoint.
5. Terminate MACD by typing!
6. Activate program to be debugged with OPCOM, for example using the RT-command.
7. When the breakpoint is reached, MACD is activated and the user may proceed as usual..

4.4.4 Breakpoint

A breakpoint is set by typing the point (.):

< expression > .

MACD will note that a breakpoint is set in the current coreload with the address given by < expression>.

If < expression > equals zero or is non-existent, no breakpoint is set. To continue the execution of a user program after a breakpoint is reached, the exclamation mark (!) is used:

< expression > !

If < expression > is zero or non-existent, the restart address is last breakpoint reached.

If < expression > equals one (1!) the last breakpoint is advanced one location and the user program is restarted at the last breakpoint location.

Else if < expression > is neither zero nor one it is taken as actual restart address.

)RBP is a Reset Break Point command. It causes MACD to forget all about breakpoints and restart address. When < expression > ! is given, the previous use of)RBP causes the user program to be terminated (RTEXIT).

4.4.5 Illegal Entry to MACD

If a jump is made to location zero in resident core, MACD will be activated as if a breakpoint was reached. However, if no breakpoint was set, an error message is generated:

ILL ENTRY ADR ZERO, RT = dddddd

where dddddd is the RT-description address of the RT-program currently active. The RT-program is then terminated immediately.

4.5 RT-Loader

4.5.1 Introduction

The RT-Loader is designed to operate with a mass storage version of the SINTRAN Real Time Operating System. Its main function is to load program units in Binary Relocatable Format (BRF) while the system is on-line.

The functions of the RT-Loader may be summarized as:

- Relocate the program unit so that the code conforms to the specific locations in core memory where it is loaded or later is to be executed.
- Link the program units together by means of linking table containing symbolic names of entry points.
- Maintain a symbolic file (RTFIL) on mass storage containing the names of all real time programs known to the system.

- Allocate the generated real time program descriptions to be used by the SINTRAN MONITOR.

- Print out of linking tables and the RTFIL file.

- Execute "editing" functions specified by the operator on the linking tables, the RTFIL file and the core loads (mass storage areas containing programs brought into core memory for execution).

- Allocate data storage in resident core and on the core loads.

4.5.2 Operation

The RT-Loader is called for executions with the SINTRAN Operators Communication Program.

The first time the RT-Loader is executed after having been installed an initial dialogue with the operator takes place. In this conversation the areas of resident core to be used for programs and data for real time program descriptions are determined.

Whenever the RT-Loader starts executing, it demands to know which core load to operate upon. This is done by an initial question: CORELOAD? which the operator must respond to by typing a valid coreload number.

The normal way of operating the RT-Loader is to specify commands in a conversational mode on the Teletype. The operator specifies one of 29 commands, the RT-Loader executes the specified function and requests another command from the user.

4.5.3 Requirements

The RT-Loader is used in a SINTRAN mass storage system (drum or disk). For effective use it will require a high-speed input device such as paper tape or card reader. The RT-Loader itself requires approximately 4K of core storage and should preferably be located on a coreload. It will use a SINTRAN on-line file with at least the size of 4 x (N + 1) words, where N is the maximum number of real time programs to be included in the system. The SINTRAN Operators Communication Program should be generated to contain the command LOAD (library symbol: LODIN).

The RT-Loader sets two requirements on the layout of core memory:

i) Stack area of SINTRAN must be below the area of resident core to be used for RT-descriptions and resident programs and global common data.

ii) Coreload area must be just above the RT-Loader area defined above (i).

The core layout is illustrated in Figure 4.1.

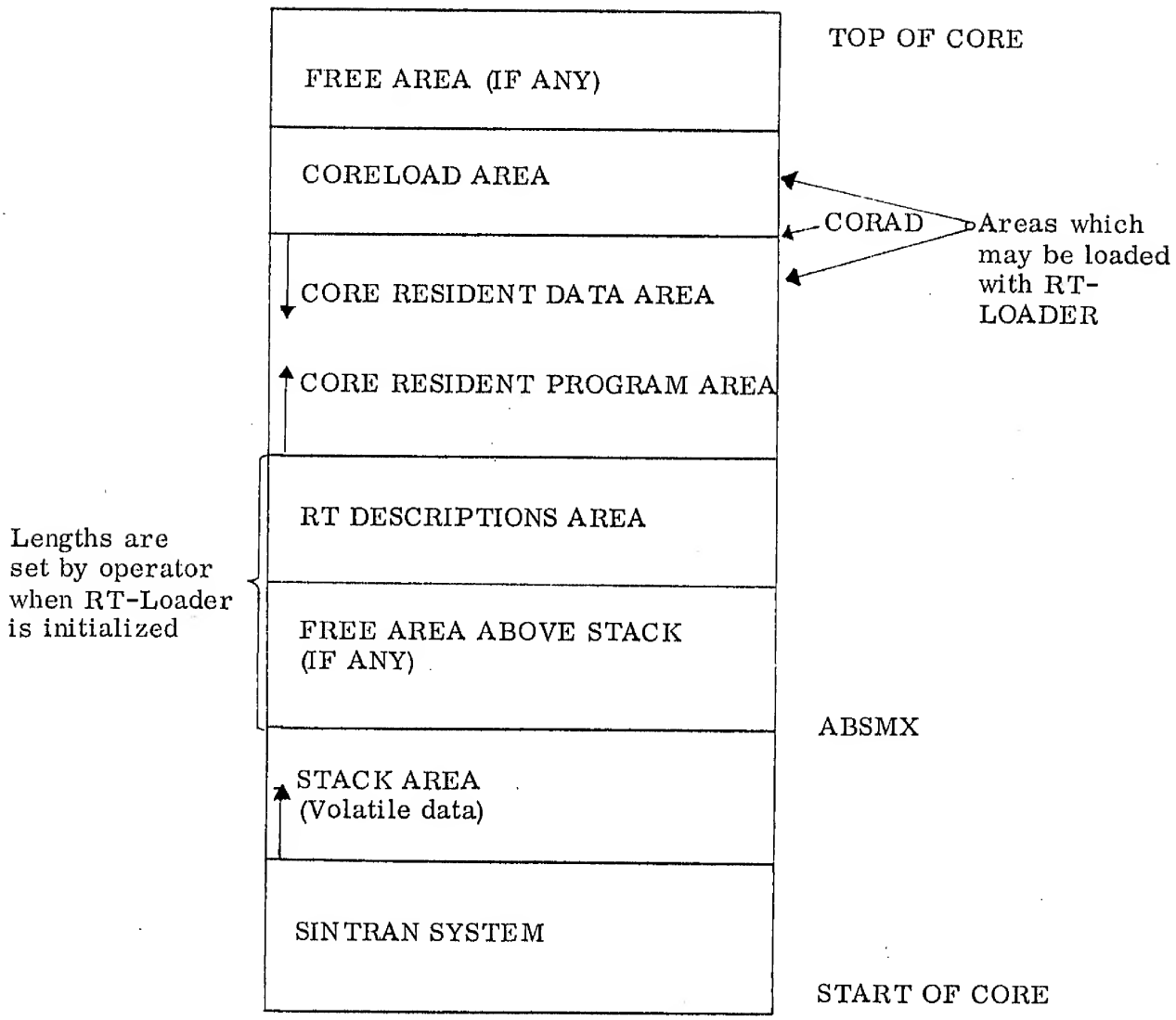


Figure 4.1 Layout of Resident Core

4.5.4 How to use the RT-Loader

This section describes how the RT-Loader is scheduled for execution and the initial dialogue taking place the first time the RT-Loader is used after having been installed.

Start up

The RT-Loader is scheduled for execution using the SINTRAN Operator Communication Program with the LOAD command:

```
@ * LOAD
```

When the RT-Loader is ready an identifying message is printed:

```
RT-Loader xxyy
```

where xxyy is a version number (xx = last two digits of the year and yy = month).

It is possible to call up the RT-Loader from any Teletype enabled for operator communication. However, only one user may use it at the same time. In fact, only one user may use anyone of the following programs at the same time:

MACD Assembler, RT-FORTRAN Compiler
RT Loader or Conversational Editor.

Initializing first time

The first time the RT-Loader is executed, an initializing procedure is performed. The RT-Loader will require answers to two questions:

i) FREE AREA ABOVE STACK?

The octal number given as answer is added to the address of the top the stack area giving the start address of the area to be reserved for real time program descriptions (see the Figure 4.1).

The resulting address must be below the coreload area. Usually the free area thus reserved is of zero length, however, a specific system might wish to set aside part of resident core for future modifications.

iii) NO OF RT-NAMES?

The octal answer determines the area to be reserved for real time program descriptions. Thus the answer should be the intended maximum number of real time programs to be loaded into the system.

The size of the area will be nine times the number of RT programs plus two. The area will start at the address previously calculated and should not extend into the core-load area.

Both answers must be typed in octal notation and are terminated with a non-alphanumeric character. The RUBOUT key may be used as an escape character if something wrong was typed, causing the complete initializing procedure to restart.

If the answers are not approved of, the error message:

ILL PAR

is printed and the complete initializing procedure is restarted.

The example below shows the RT-Loader being called, the initializing procedure, and a printout of the available load areas (WL).

Sixteen words decimal are set aside above the stack which ends at 17000. The RTFIL is initialized to hold a maximum of 51 real time programs.

```
@ * LOAD

RT-LOADER 7106
FREE AREA ABOVE STACK? 20
NO OF RT-NAMES? 63
CORELOAD? 5

* WL
CORE  = 017736 - 020000
CLOAD = 020000 - 040000

*
```

A rough description of the initializing procedure itself follows: Two tables are initialized to contain the lower and upper addresses of all areas available for loading (i.e., resident core and the core-loads). The RTFIL and mass storage are initialized with the addresses of all available RT-descriptions. If the file is too small, the run time error RUN ERR 51 occurs and the RT-Loader program is aborted. The area reserved for real time program descriptions is cleared.

The linking table is initialized to contain as permanent entries those SINTRAN MONITOR entry points which are located in resident core and which are defined in the specific MONITOR configuration presently used.

At last the RT-Loader itself is changed so as it circumvent the initializing procedure next time it is called and the control is passed to the conversational part to receive a command from the operator.

4.5.5 Command Use

In general the RT-Loader signals it readiness for reading a command line by printing an asterisk (*) to the left on the line. The operator must type a command line consisting of two alphabetic characters followed by any parameters required. As separators between the command name and the parameters, and as command line terminator any non-alphanumeric character may be used (the space character for example). The RUBOUT key may at any time be used as an escape character to delete the current line being typed.

Coreload?

However, before the RT-Loader requests commands it must know which part of the mass storage unit it is going to load into. That is, it requires an answer to the following question to when it starts executing or when the RT-Loader is explicity reset (*RS, se below):

CORELOAD?

The operator must type back the number of the coreload to be loaded or the number 0 (zero). The coreload number must be in octal notation and must denote a coreload available to the RT-Loader (limits on the coreload numbers are set when the RT-Loader is assembled). An answer of 0 (zero) is treated as a special case. This answer indicates to the RT-Loader that only resident core is to be loaded. Thus, the loading commands operating on mass storage (MM and AM) will be illegal. And, any common areas defined during the following loading operations are automatically made core resident and the corresponding common area labels are made permanent symbols.

4.5.6 RT Loader Command Summary

Command Name	Parameter(s)	Description
AC	-	Autoload Core
AM	-	Autoload Mass Memory
CC	coreload number	Clear Core
CL	common area label	Declare Common Label area in resident core
DE	entry point name	Delete Entry point
DP	RT-program name	Delete RT-program
EN	-	End load; include what was loaded
EQ	entry point name and address	Define entry point (EQuality)
ER	number of definitions	Equality Read } EQ's from
ES	number of definitions	Equality read Selectively } input device
EX	-	Exit the RT-Loader
FX	-	FIX, making Linking Table permanent
IN	input device	Select input device
LP	-	Enable Label Printout (RT-FORTRAN)
MC	-	Manual Load Core
MM	-	Manual Load Mass Memory
PD	output device	Select output device for next W command
RF	library symbol name	Make reference to a library symbol
RS	-	ReSet the RT-Loader
RT	RT-program name	Declare RT-program name
WA	-	Write all symbols in Linking Table
WC	-	Write Common area labels
WE	-	Write non-permanent Entry points
WF	-	Write permanent (Fixed) entry points
WG	-	Write Global variable names
WL	-	Write Location counters
WP	-	Write RT-Program names
WR	-	Write Referenced entry points
WS	-	Write descriptor referenced entry points.

4.5.7 Binary Relocatable Format Code

Control Byte (8 bits)	Mnemonic	Parameters (16 bits)	Action
0	FEED	0	Neglected before BEG and after END, else illegal.
1	LF	1	$W_1 \rightarrow ((CLC))$, $(CLC) + 1 \rightarrow (CLC)$
2	LR	1	$W_1 + (PB) \rightarrow ((CLC))$, $(CLC) + 1 \rightarrow (CLC)$
3	LC	(1)	ILLEGAL
4	AFF	2	$W_1 + (W_2) \rightarrow (W_2)$
5	ARF	2	$W_1 + (PB) + (W_2) \rightarrow (W_2)$
6	ARF	2	$W_1 + (W_2 + (PB)) \rightarrow (W_2 + (PB))$
7	ARR	2	$W_1 + (PB) + (W_2 + (PB)) \rightarrow (W_2 + (PB))$
10	SFL	1	$W_1 \rightarrow (CLC)$
11	AFL	1	$W_1 + (CLC) \rightarrow (CLC)$, store zeroes
12	SFL	1	$W_1 + (PB) \rightarrow (CLC)$
13	COMN	(1)	ILLEGAL
14	MAIN	2	Define name of main program Start = (CLC)
15	LBR	2	Library program unit
16	ENTR	2	Define entry point name, address = (CLC)
17	BEG	0	Start of program unit
20	REF	2	Reference to an external name
21	END	1	End of program unit, W_1 = checksum
22	INHB	0	Illegal tape, source code contained errors
23	EOF	0	End of several program units
24	LNF	$1 + W_1$	$(W_i \rightarrow ((CLC)), (CLC) + 1 \rightarrow (CLC))_{i=2}^{1+W_1}$
25	PRIOR[1)]	1	Define priority for previous main program
26	ASF	3	Define common area label and length
27	ADS	2	Add value of common label to (CLC)-1
30	AGL	3	Define global (area) label and length
31	ADG	2	Add value of global label to (CLC)-1
32	DREF		ILLEGAL

CONT.

[1)] The PRIOR-byte is supposed to follow the MAIN-byte.

Control Byte (8 bits)	Mnemonic	Parameters (16 bits)	Meaning
33	RUT	$1+W_1$	Check of subroutine calls
34	INL	2	Integer local data
35	DBL	7	Double local data
36	RLL	4	Real local data
37	RXL	7	Complex local data
40	INC	4	Integer common data
41	DBC	9	Double common data
42	RCL	6	Real common data
43	CXC	9	Complex common data

4.5.8 Error Diagnostics

Loading Errors

The RT-Loader makes several error checks when it processes the binary relocatable code. When an error is detected, the RT-Loader prints an error message. The format of the error message is

ERROR CODE IN SYMBOL1 SYMBOL2

where ERROR CODE is a two letter code which specifies the type of error, SYMBOL1 is the name of the current program unit, and SYMBOL2 will only occur with the first four error codes listed below and it will give further indication of the error cause.

The RT-Loader's error messages are listed and explained below:

Error Code	Explanation
DF	Data Error - The program unit being loaded has either tried to extend a previously defined common area or tried to alter the definition of previously undefined global variable. SYMBOL2 is the common area label or global variable affected.
UD	Undefined Data - The program unit loaded refers to either a common area label or a global variable name which has not previously been defined. SYMBOL2 is the name in question.
DD	Double Definition - This is a common error message for several errors: i) Real time program name equals the name of an entry point. ii) Two entry points have the same name. iii) Entry point name equals the name of a real time program. SYMBOL2 gives the offending symbol name.
LF	Loading Area filled - The current area being loaded is filled; either causes by program instructions overflowing the data area, or the data area being extended below the upper limit of loaded program instructions. In the last case, the data area name is printed as SYMBOL2.

Error Code	Explanation
CS	Check Sum - The checksum on the input tape does not match the checksum computed during loading. This may indicate a faulty input device or a torn tape. It may also indicate an error with the device used to produce the tape.
EP	Error Program - The program unit loaded contains source language errors. That is, either an RT-FORTRAN program error detected by the compiler or an assembly program error detected by the MAC assembler.
IB	Illegal Control Byte - A BRF control byte did not correspond to the allowed ones. This may be caused by having used a wrong version of the compiler or by wrong reading of the input tape.
TF	Tables filled - The internal RT-Loader tables were filled. Either reconfigure your version of the RT-Loader to have more room for tables (assembly time parameter FLDNG rewrite your programs with fewer program units and/or labels.
SF	System filled - The maximum allowed number of real time programs has been included. There are no more real time program descriptions available.
PR	Priority wrong - The priority specified was greater than 255 in the real time program being loaded.
IL	Internal RT-Loader Error - These may be caused by a bug in the RT-Loader or a fault in the computer hardware itself.
AD	Address - A program relative address of zero was specified in the BRF code of the program being loaded. This may be caused by having used an old version of the assembler or compiler which produced BRF code relative to location zero (or as experience has shown, by having used the assembler in an inappropriate way).

4.5.9 Miscellaneous RT-Loader Printouts

	Explanation
CORELOAD?	RT-Loader demands the coreload number on which to operate.
DD!	Double definition of entry points and/or RT-programs while using certain commands (EQ, ER, ES, RT).
FREE AREA ABOVE STACK?	Initializing procedure question.
ILL PAR	Error printout in initializing procedure.
NO PRIOR IN <program name>	Warning message. RT-program beeing loaded does not specify its priority.
NO OF RT-NAMES?	Initializing procedure question.
OK!	The EN command was successfully executed.
PREVIOUS ERROR	The RT-Loader has previously detected a loading error and will not execute the EN or loading commands.
REF'S!	The EN command is neglected due to undefined references not being satisfied.
REPLACING <program name> ?	RT program being loaded has the same name as an existing one. The operator is questioned whether this is a replacement or not.
RT LOADER	Identifying message printed when the RT-Loader is started.
TOO LONG	Symbolic command parameter has more than five characters.
%LABELS IN <program name>	Heading printed in the response of the LP command after having loaded one program unit.
?	Common error indicator for several commands.
*	RT-Loader requests a command from the operator.

4.6 RT FORTRAN Compiler

4.6.1 How to use the Compiler

When the user wants to compile and run his source program, he should follow the procedure given below:

a) Use the operator command FTN.

b) The compiler answers with C*.

c) The user must now specify the devices to be used in the following order: Source input device, list device, and object output device. Each device is typed on the Teletype as a decimal number terminated by comma or space and comma. The record is terminated by carriage return. If the device number is greater than 8, the compiler assumes that the appropriate data is found/placed on a file. For numbers from 0 to 7, inclusive, the user must supply the standard device numbers as specified by ND. Thus, it would be illegal to use device 4 (card reader) as an output device. If the decoding routine finds any error, it writes

```
ERR F00
C*
```

and the user should try again. For instance, the following commands would be legal:

```
4,0,3↵
1,0,0↵
2,5,0↵
```

The first one means: Read source from card reader, do not output any listing and output object code on the punch.

The second one means: Type program on Teletype, do not output any listing or object code.

The last one means: Read source program from tape reader, output listing on line printer and do not output any object code.

d) When the sequence is accepted, the compiler types back C*. The user should now specify A↵ if his source program ends with EOF or M↵ if he only wants to compile one program unit.

The compiler will start compiling the source program from the input device, list on the list device, and output BRF code on the object device.

COMPILER ERROR MESSAGES LISTED BY NUMBER

01	Unrecognizable statement.
02	Illegal left side of an arithmetic statement.
03	Symbol contains more than five characters.
04	Syntax error in a DIMENSION statement.
05	Variable in a DIMENSION statement invalid or previously dimensioned.
06	Incorrect subscript in a DIMENSION statement.
07	Syntax error in a COMMON statement
08	Variable in a COMMON statement invalid because it was previously COMMON or is a subprogram formal parameter.
10	Syntax error in a DO statement.
11	DO statement label previously defined.
12	Invalid DO control variable.
13	One of the parameters I, M2 or M3 may be changed within the DO loop (DO 1 I = M1, M2, M3).
14	Improperly nested DO loop.
15	DO loop ended with a transfer statement, STOP statement or DO statement.
16	More than 10 levels in a nested DO loop.
17	Syntax error in a GOTO or computed GOTO statement.
18	Transfer (reference) to itself in a GOTO statement, computed GO TO, IF or I/O statement.
19	Invalid or missing variable in a computed GOTO statement.
20	Syntax error in an IF statement.
21	Syntax error in an arithmetic expression.
22	Missing operator or left parenthesis in an arithmetic expression.
23	Missing operand in an arithmetic expression.
24	Illegal use of a SUBROUTINE name.
25	Unbalanced parentheses.
26	Illegal character.
27	Control transferred to a FORMAT statement or label in an I/O statement does not refer to a FORMAT statement.
28	Zero used as label number.
29	Same statement number assigned to more than one statement.

30	Labeled END statement
31	Undefined label.
32	Statement following a transfer statement must be labeled.
33	RETURN statement appeared in a main program.
35	Incorrect or too big constant or label number.
36	In a FORMAT statement the first character is not left parenthesis and the last character is not right parenthesis.
38	Variable FORMAT reference not an array name.
39	Unlabeled FORMAT statement.
40	Attempt to call a subprogram recursively.
41	Syntax error in a PROGRAM statement.
42	Syntax error in a SUBROUTINE or FUNCTION statement.
43	More than 31 (30) parameters in SUBROUTINE or FUNCTION statement.
44	No argument in a FUNCTION statement.
45	Repeated argument in a SUBROUTINE or FUNCTION statement or the subprogram name used as parameter.
46	Real expression in an array subscript.
47	More than two array subscripts.
48	Illegal actual parameter.
49	Syntax error in an I/O statement.
50	Statement contains redundant characters.
51	Missing END statement.
52	Specification statement labeled or preceded by an executable statement.
53	Syntax error in a CALL statement.
54	Function gets no value in a FUNCTION subprogram.
55	Syntax error in an EXTERNAL statement.
56	Syntax error in a GLOBAL statement.
60	Warning that the function value might not be defined when returning through a RETURN statement.
95	Warning that the first local block is filled up.
96	Too complicated expression for compiler tables.
98	Symbol table full.
99	Compiler destroyed or may be a bug in the compiler.

Any error messages will be typed on either the list device or, if this is zero, on the Teletype, and the source stream will stop when the compiler finds the next END statement.

When the compiler terminates reading the source program after finding END for M or EOF for A, it writes C* again. It is now ready to read new source programs. The compiler will remember the old device definitions and the user may, if he wants, simply type A↵ or M↵. Initially the devices are card reader for input device, zero for listing device, and paper punch for output device. This is equivalent to starting the compiler and typing 4,0,3↵.

e) When the user is finished compiling, he must type H↵. Then the compiler will be terminated (RTEXT).

f) The object tapes produced by the compiler can now be loaded by the RT-Loader.

4.6.2 Compiler Error Messages

The error message will be written on either the line printer or the Teletype, depending on which is specified as the listing device. If the user has requested a listing of the program, error messages will be printed on the line following the erroneous line and, in certain cases, on the next line thereafter.

The error messages have the format:

ERR Fdd

if the error is in the program line above, and

**ERR Fdd

if the error is in the next to the last line. The error number dd is found in the error message list. On the line before the error message is printed, an "↑" points at the character where the error was found. If the program is not listed on the Teletype or on the line printer, the erroneous line is indicated in the format

11111+ccccc

where 11111 is a label number, and ccccc is the number of statements beyond the 11111. The line before the first statement is indicated by 11111 = 0. The full error message will then be

11111+ccccc ERR Fdd

or

11111+ccccc **ERR Fdd

4.7 Using the MACM Assembler

4.7.1 Purpose

MACM is a modified standard MAC assembler. The main difference is its ability to assemble programs out on a mass storage device (drum) in a core image format.

When MACM is working it has complete control of CPU and external devices.

MACM has also the capability to swap itself with the core image on mass storage and start in a specified location. Used together with the two program tapes, User Break Point (UBP) and Core to MACM transfer (CTOM), the user may freely change between the MACM assembler and his own program much the same way as with an ordinary MAC assembler. However, the problem of protecting MACM from the user program is non-existent and all core is available to the user.

As an aid to debugging MACM has been equipped with commands to save ( )GJEM) and restore ( )HENT) the current core image to and from a save area on the drum. The saved area may be compared with the core image area word by word by means of the compare command (a < b;)SAM). The core image area may be loaded from core using CTOM after having run the user program.

The contents of a coreload may be moved to another coreload; ( )CMOVE).

A rebootable tape may be punched out using the core image as source ( )BPUND).

When this tape is placed in the reader and the LOAD button pushed, it will automatically be read onto the mass storage.

An additional mode, "system definition mode", may be used when assembling application programs.

This mode ensures that only those system symbol defintions which are referred to will be taken care of, others are ignored, when reading in a list ( )LIST) of definitions from a system or main program.

In order to use MACM to link programs separately assembled, the undefined reference list may be punched out ( )ULIST).

4.7.2 MACM Commands

)GJEM

copies the complete current core image to a save area on the drum. Current coreload number is saved for later use by)HENT.

)HENT

restores the core image from the save area, using the coreload number saved by)GJEM.

)SAM

compares word by word the core image area and the save area. Any differences are printed on the Teletype. The)SAM command is used with lower and upper limits for the compare:

```
A  B
)SAM
```

Words from address A to address B both inclusive are compared.

)CMOVE N1 N2

moved (copies) the contents of coreload number N1 into coreload number N2.

)BPUND

punches a rebootable tape of the area defined by lower and upper limit (A < B) in the current core image. When the tape later is to be loaded, the hardware device number of the mass storage must be placed in location 0.

)CLOAD N1

defines the current coreload number, N1, to be used by MACM henceforth.

The number N1 is checked to see if it is within allowable limits (≥ 0 and $\leq$ CLM), and further, if there are undefined symbols in the table, the warning message "UNDEF SYMBOLS" is printed. This is due to the fact that MACM does not know in which coreload undefined symbols are referred. - Thus it is the responsibility of the user to have the proper coreload number set when symbols which have been referred are defined.

The symbol, C, will always have a value equal to current coreload number. To have the current coreload number printed out, C: is typed on the Teletype and MACM will print out the number.

)FIX

makes all symbols in the Local Symbol Table permanent. That is, they will not be deleted by any)CLEAR command given later on.

This command is intended to be used to make global (or system) variables permanent when assembling a system from parts.

)SYSDF

puts MACM in a System Definition Mode. This mode is reset by)LINE (@).

While in System Definition Mode only those symbols referred to in the Undefined Symbol Table will be defined when inputting an equal sign definition to MACM. All other equal sign defintions are ignored.

The purpose of this mode is to avoid filling up the tables with unnecessary symbols.

)ULIST

is a PUNCH command making it possible to link several separately assembled, but interrelated programs using the assembler.

)ULIST outputs the Undefined Reference Table in symbolic code with the following format:

< octal address > / ↑ < undefined symbol name > % previous contents in this location

Proposed use:

Each program part is separately assembled and)LIST,)ULIST and)BPUND tapes are produced for each part. Finally the different parts are linked together by the following three step procedure:

1. Load all binary tapes
2. Input all)ULIST tapes to MACM
3. Input all)LIST tapes to MACM

The System Definition Mode ( )SYSDF) may successfully be used in step 3.

)LINE or @

In addition to the previous definition of this command, "return input control to the Teletype", the following will also be executed:

1. Reset System Definition Mode (see)SYSDF)

and

2. Update core image on mass storage by emptying the mass storage block buffer in MACM.

)XPUN

This command is included in order to punch out a standard binary tape of MACM itself.

)XPUN may punch out several disjoint areas of core:

```
a < b
)XPUN↵

c < d
)XPUN↵

e < d
)XPUN↵
```

Finally the lower and upper limits are zeroed and the)XPUN command is followed by the symbolic start address:

```
0 < 0
)XPUN  MAC↵
```

)9READ

reads binary tape produced by the)BPUN command into the current core image. The octal part is ignored, i.e. the command searches for the exclamation mark.

The contents of the binary tape are placed on the core image between the limits specified for)BPUN.

Note that the command takes input from the device specified for object output! Remember to reset this device number immediately after the)9READ command.

)9BYTT < Ten Symbols separated by Space >

This command makes it possible to change the "basic parameters" of a MACM system. This means, for example, that the same binary version of MACM may be used for drum as well as disc. The ten symbolic parameters for)9BYTT must be previously defined. The meaning of the parameters is explained below:

MSTYP	=	Mass storage type. Drum =.0, disc = 1
DEVNO	=	Primary mass storage device number
CORAD	=	Start address of coreload in core
LONG	=	Length of coreload in words
CLM	=	Upper limit for coreload numbers (inclusive)
BLST	=	"Mass storage address" of coreload number one
DRES	=	"Mass storage address" of core resident core image
CRMAX	=	End of core address (37777 for 16K core)
MACAD	=	"Mass storage address" of area where MACM is saved
DASA	=	"Mass storage address" of "GJEM/HENT" area.

After the symbols have been given the desired values, type the command:

)9BYTT ␣MSTYP ␣DEVNO ␣CORAD ␣LONG ␣CLM ␣BLST ␣DRES ␣CRMAX ␣MACAD ␣DASA

MACM now writes CR-LF indicating that the command has been executed. If a symbol in the parameter string is not defined, the error message:

FABS NOT FOUND

is printed. This means fixed absolute symbol does not exist in MACM's symbol table. Restart with)9BYTT.

The symbol names may of course be anything, but the order of the parameters is essential (as described).

)9CTOM

is connected to the object stream. It produces an octal tape described in Section 5.2. It is important to remember that some parameters given to)9BYTT are used. Thus a "CTOM-tape" must always correspond to the (latest) 9BYTT command.

The ordinary breakpoint commands in MAC are applicable:

<address expression>.	Set breakpoint at specified location.
1!	Execute one instruction by advancing the breakpoint.
!	Continue from last breakpoint (may even be used without changing last breakpoint location).
)RBP	Forget all about breakpoints.
0.	Set no breakpoint.
<address expression>!	Start user program in specified location.

Note! Breakpoints may only be used with the core resident core image area (no breakpoint may be set on coreloads 1, 2.....).

4.7.3 Debugging and running User Programs

Start User Program

Transferring control from MACM to a user program is done the usual way by writing a start address followed by the exclamation mark, <Defined expression>!. This will cause MACM itself to be saved on the drum in a MACM save area. The current user core image is read into core and control is transferred to the specified location.

NOTE: The upper 28 locations of the user core image should not be used as these are used by the transfer routine.

Using start address 0 (zero) will not cause a transfer to location zero, but rather force a JMP * to be executed in the transfer program when swapping is finished.

Return to MACM

When the user has tried executing his program, he may want to return to MACM either to make corrections in his current program or for reload of core of a fresh program.

This may be done with a 2-part program tape in octal format read by the hardware assembler called CTOM (Core TO MACM).

The first part is used if the user wants to save his current core status, i.e. core is written on the core resident core image area of the drum.

The second part causes MACM to be loaded into core from its save area and started.

These two parts of the CTOM tape are separated by some blank tape so as the user may only use the second part if he desires. However, this will cause his current core status to be lost.

Breakpoint

In order to make use of the breakpoint function a User Break Point tape (UBP) must be assembled together with the user program. UBP will cause an automatic core-swapping and start of MACM whenever a breakpoint is reached via location 0 and 2.

The ordinary breakpoint commands in MAC are applicable:

<address expression>	Set breakpoint at specified location.
1!	Execute one instruction by advancing the breakpoint.
!	Continue from last breakpoint (may even be used without changing last breakpoint location).
)RBP	Forget all about breakpoints.
0.	Set no breakpoint.
< address expression>!	Start user program in specified location.

Note! Breakpoints may only be used with the core resident core image area (no breakpoint may be set on coreloads 1, 2 .....).

4.8 Using the Conversational Editor

When the Editor is started, it asks for some options, usually being answered with Y.

The following extra commands for use under SINTRAN are implemented:

control/C	Terminate the Editor; return to SINTRAN. (Type the character two times.)
control/P	Break the imput/output operation going on. (Return to the Editor in command mode.)
m,nU↵	Change the logical unit numbers used by the R and P commands. m = input unit n = output unit (Initial values: m = 2 and n = 3.)

4.8.1 Summary of Conversational Editor Operations

Special Key Functions:

Carriage Return (RETURN Key)	Text Mode	- Enter the line in the text buffer.
	Command Mode	- Execute the command
Back Arrow (←)	Text Mode	- Cancel the entire line of text, continue typing on same line.
	Command Mode	- Cancel command. Editor issues a ? and carriage return/ line feed.
Rubout (\)	Text Mode	- Delete from right to left one character for each rubout typed. Does not delete past the beginning of the line. Is not in effect during a READ command.
	Command Mode	- Same as back arrow.
Form Feed (CTRL/ FORM Key Combination)	Text Mode	- End of inputs, return to command mode.
Period (.)	Command Mode	- Current line counter used as argument alone or in combination with + or - and a number (.,.+5L).
Slash (/)	Command Mode	- Value equal to number of last line in buffer. Used as argument (/-5,/L).
Line Feed (↓)	Text Mode	- Used in SEARCH command to insert a CR/LF combination into the line being searched.
	Command Mode	- List the next line (equivalent to .+1L).
ALT Mode (ALT key) Escape (ESC key) Right Angle Bracket (>)	Command Mode	- List the next line (equivalent to .+1L).
Left Angle Bracket (<)	Command Mode	- List the previous line (equivalent to .-1L).
Equal Sign (=)	Command Mode	- Used in conjunction with . and / to obtain their value (.=27).

Colon (:)	Command Mode	- Lower case character, same function as =.
Tabulation (CTRL/TAB Key Combination)	Text Mode	- Produces a tabulation which on output, is interpreted as spaces or a tab character depending on the "TAB PUNCHING"-Editor Option.
CTRL/C (two times)	Return	- Exit from the Editor. SINTRAN Operator Com. can be used.
CTRL/P	Break	- The input/output going on will be terminated, setting the Editor in command mode.

Initializing:

Question	OPR-bit	Position	Meaning
-	15	0 (N) 1 (Y)	Normal operation Suppress output
HIGH SPEED INPUT	14	0 (N) 1 (Y)	Low speed input High speed input
HIGH SPEED OUTPUT	13	0 (N) 1 (Y)	Low speed output High speed output
PARITY INPUT?	12	0 (N) 1 (Y)	Neglect parity on input Test parity on input
PARITY OUTPUT?	11	0 (N) 1 (Y)	Do not generate parity Generate equal parity
SPACES TO A TAB WHILE READING?	10	0 (N) 1 (Y)	Read input tape as it is Convert spaces to a tabulation on input
TAB PUNCHING	9	0 (N) 1 (Y)	Output tabulations as spaces Output tab characters

Command Summary:

Command	Format(s)	Meaning
READ	R↵	Read incoming text from reader and append to buffer until a form feed is encountered.
	nR↵	Read incoming text until n lines have been read.
APPEND	A↵	Append incoming text from keyboard to any already in buffer until a form feed is encountered.
LIST	L↵	List the entire buffer.
	nL↵	List line n.
	m,nL↵	List lines m through n inclusive.
PUNCH	P↵	Punch some blank tape, the entire buffer, a form feed and some blank tape.
	nP↵	Punch line n.
	m,nP↵	Punch lines m through n inclusive.
TRAILER	T↵	Punch four inches of trailer.
NEXT	N↵	Punch the entire buffer and a form feed, kill the buffer and read the next page.
	nN↵	Repeat the above sequence n times.
KILL	K↵	Kill the buffer.
DELETE	nD↵	Delete line n of the text.
	m,nD↵	Delete lines m through n inclusive.
INSERT	I↵	Insert before line 1 all the text from the keyboard until a form feed is entered.
	nI↵	Insert before line n until a form feed is entered.
CHANGE	nC↵	Delete line n, replace it with any number of lines from the keyboard until a form feed is entered.
	m,nC↵	Delete lines m through n, replace from keyboard as above until form feed is entered.
MOVE	m,nØkM↵	Move lines m through n inclusive to before line k.

Command	Format(s)	Meaning
GET	G↵	Get and list the next line beginning with a tag.
	nG↵	Get and list the next line after line n which begins with a tag.
SEARCH	S↵	Search the entire buffer for the character specified after the carriage return. Allow modification when found.
	n S↵	Search line n, as above, allow modification.
	m,nS↵	Search lines m through n inclusive, allow modification.
UNIT	m,mU↵	Change the logical unit numbers used by the R and P commands m = input unit n = output unit.

5 ERROR SITUATIONS

5.1 Error Messages

At run time, errors may be detected by the system. Most of the error will cause the current RT program to be aborted and the error message:

aa.bb.cc RUN ERR nn xx yy

to be printed on the system Teletype. The parameters should be decoded as follows:

aa.bb.cc - Time when the error message was printed
  aa - hours
  bb - minutes
  cc - seconds

nn - Error number. For further explanation see Section 5.4.

xx - Usually the name or octal address of current RT-program where the error was detected. For some of the error numbers it will be an octal number with a different meaning.

yy - Decimal number whose meaning is different to the different error numbers.

xx or yy will be omitted for some error numbers.

Example: 17.30.00 RUN ERR 04 PROGA

If more than two errors occur at the same time, the last message will not be reported. However, the message "RUN ERR" without parameters will be printed, indicating that some error message is lost.

In case of a mass storage transfer error, an additional message TRANSF! will be given.

5.2 Program Errors

Program errors will be reported if for instance a monitor subroutine is called with an illegal parameter value. Usually the failing program will be aborted immediately, and the proper error message will be printed.

5.3 Peripheral Device Errors

If a transfer has not been completed within a given time, the message RUN ERR 37 will be given, followed by the name or octal address of a possible waiting RT-program and the decimal logical unit number. The RT-program will be set in a waiting state, and the operator may use one of the commands

ABORT, PIN, PINC, or TERM (see Section 4.3.2)

Other peripheral device errors will give immediate aborting of any program waiting for the device.

5.4 Run Time Errors

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
02	Illegal formal type (possible system error)	RT-prog.		Yes	RT-FORTRAN Run time system
03	Wrong number of parameters	"		"	"
04	Disagreement between actual and formal parameter	"		"	"
06	Illegal actual type in subroutine call (possible system error)	"		"	RT-FORTRAN Run time system I/O
07	Illegal block in parameter descriptor	"		"	RT-FORTRAN Run time system
09	Parameter value on coreload by call of a subroutine on a different coreload	"		"	"
10	The sobroutine has tried to modify a constant parameter	"		"	"
22	Wrong delimiter in READ/WRITE statement (error in compiled code)	"		"	RT-FORTRAN Run time system I/O
27	Wrong checksum on binary tape	"		"	Operator Communication
28	Illegal variable or format specification	"		"	ROUT
29	Parity error in ASCII input	"		"	"
30	Illegal logical unit number	"	Logical unit no.	"	SINTRAN I/O

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
33	Interrupt hang-up on I/O device	Hardware device no.	Logical unit no.	No	SINTRAN I/O
35	String loo long	RT-prog.	"	"	"
37	Missing interrupt from I/O device (time-out)	RT-prog. waiting for input	"	"	SINTRAN I/O (timer)
38	Card reader error		"	"	SINTRAN I/O
39	False interrupt from I/O device	Hardware device no.	"	"	"
48	Not loaded RT-program called for execution	RT-prog.		Yes	RT-Loader
49	Not loaded sub-program called	"		"	"
50	Drum transfer error	"	Block no.	"	Monitor
51	Parameter error in a call of TRANS	"		"	"
52	Illegal time unit in SET, INTV, CLADJ or TIME	"		"	"
53	Stack overflow, or negative parameter in call of STACK or PUSH	"		"	"
54	Negative or too great parameter in call of UNSTK. The call is ignored	"		No	"
55	Priority out of range in call of PRIOR. The call is ignored	"		"	"
56	Interrupt line number out of range in call of CONCT. The call is ignored	"		"	"

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
57	Wrong B-register contents in call of POP or PO	RT-prog.		Yes	Monitor
58	Parameter error in call of INHIB	"		"	"
59	Memory protect violation	"	P-reg. contents	"	"
60	Insufficient stack space before call of PUS	"		"	"
61	Coreload number too big	"		"	"
63	Parameter error in call of UPDAT	"		"	"
64	Parameter error in call of ABSET	"		"	"
65	Attempt to put an RT program = 0 into time or execution queue	"		No	"
66	Unidentified interrupt		Hardware interrupt level	"	"
71	Illegal character in format	"		Yes	FORTRAN I/O
72	Parenthesis nested deeper than 2	"		"	"
73	Attempt to fetch character beyond format	"		"	
74	Attempt to store character beyond format	"		"	"
75	Illegal device number	"		"	"

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
76	Argument error, unidentified type specification	RT-prog.		Yes	FORTRAN I/O
80	Trying to store characters beyond buffer	"		"	"
81	Trying to fetch characters beyond buffer	"		"	"
82	Buffer overflow on input	"		"	"
83	Wrong parity in input	"		"	"
84	Bad character in input	"		"	"
85	Integer overflow	"		"	"
86	Real overflow on input	"		"	"
87	Real overflow on input	"		"	"
88	Real overflow on	"		"	"
89	System error	"		"	"
90	Too big input record	"		"	"

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal	Program aborted	Detected in
AA	Error in <real> ** <real>	RT-prog.		Yes	PORTRAN Library
AI	Error in <real> ** <integer>	"		"	"
AT	Error in ATAN2	"		"	"
CH	Error in COSH	"		"	"
CO	Error in COS	"		"	"
DI	Error in integer division	"		"	"
EX	Error in EXP	"		"	"
IX	Error in <integer> ** <integer>	"		"	"
LN	Error in ALOG1 ALOG2 or ALOG	"		"	"
SH	Error in SINH	"		"	"
SI	Error in SIN	"		"	"
SQ	Error in SQRT	"		"	"

6 FINDING STATUS AND DEBUGGING INFORMATION

6.1 Facilities when the System is running

Operator commands:

E, D, DUMP, PUNCH, READ, COMP, LOGZ, LOG,
WSTK, WEQ, WTQ, WRTS, WRTT

MACD can be used to examine every location in core or coreloads.

The RT-loader can be used to list the RT-programs loaded.

Background program:

When the system is idle, it will hang in a small loop on interrupt level zero. The register values will then give some useful information:

D is incremented for each revolution in the loop.
X holds the least significant word of the internal time, thus being incremented each basic time unit.
T holds the contents of the word, whose address is found in the panel switch register. Thus all core locations can be examined by using the panel switches and the register display, provided that the CPU is not heavily occupied with RT-programs.

6.2 Core Locations of special Interest

Fixed locations:

Location	Symbol	Meaning
3	COMFL	Compare flag. A compare operation is performed after each transfer for those mass storage devices which have their bits set. Bit 0 = drum, bit 1 = disc 1, bit 2 = disc 2, bit 3 = magnetic tape 1, bit 4 = magnetic tape 2, bit 5 = core-to-core module. The eight leftmost bits determine if the coreload should be written back in case of parity error. If bit X + 8 is set, writing back will be performed for mass storage device No. X.
4	LOADI	File number for the RT program name file. This number is set by the RT loader at its first start, and is used by the Operator Communication and the error printout RT program.
5	SYS	Address for initializing program.
6	STAUT	Start of execution queue.
7	START	Start of time queue.
10	STTOP	Start of stack.
11	RTREF	First stack location of current RT program.
12	TOPRF	First stack location of the RT program at the top of the stack.
13	RTOVR	First stack location of the RT program above the current.
14	STMAX	Maximum stack address for programs with priority less than LMPRI (defined at system generation).
15	ABSMX	Maximum stack address.
16	PRILM	LMPRI, priority limit.

RT-description

The RT-description is a data element consisting of nine consecutive locations in permanent core. Each RT-program is associated with an individual RT-description.

The nine locations are used thus:

Loc. 1 : Link. This location is used for linking RT-descriptions together to form time or execution queues.

Loc. 2 : The right half-word contains the priority of the RT-program and the left half-word some status information flags.

Loc. 3 and 4 : This is a double precision number indicating the time when the RT-program is to be (or was) scheduled.

Loc. 5 and 6 : This is the time interval if the RT-program is to be executed periodically.

Loc. 7 : This contains the core address of the first program instruction.

Loc. 8 : Bits 0 -11: Coreload No. (1, 2, 3.....).
Bits 12-14: Set aside for future extensions.
Bit 15 : If set, the RT-program needs writing back.

Loc. 9 : This is a pointer to the data field of a connected interrupt line (see the manual, Generating SINTRAN II.

The flag bits in loc. 2:

Bit 15 : Periodic execution (INTV).

Bit 14 : Interrupt line connected.

Bit 13 : In time queue.

Bit 12 : In execution queue.

Bit 11 : Started (in stack).

Bit 10 : Time is specified by ABSET.

Bit 9 : Execution inhibited.

Bit 8 : Free.

The I/O System:

Name	Displace-ment	Meaning
TMSUB	-3	Entry point of a subroutine to be called in case of time-out.
TMR	-2	Timer count location. A negative number is counted up each second. If zero is reached, TMSUB is called.
TTMR	-1	A negative number (or 0) used as initial setting of TMR.
DRIVR	0	Entry point of driver routine, entered when an interrupt occurs.
STDEV	1	Entry point of a subroutine to start the device.
TRNSF	2	Entry point of a subroutine to transfer the record between user and device + buffer.
MCLR	3	Entry point of a master clear routine to clear the device buffer.
DFLAG	4	Some flags, explained below.
SAVT	5	Used as a working location for IDENT to save its T register before entering the driver.
MLINK	6	Link location for the monitor queue.
HDEV	7	Hardware device number.
ACTON	10	RT program to be started, specified by INIT or DATUT, or RTREF (stack pointer) of the waiting RT program.
ALTRN	11	Alternative logical unit, being used if current unit is marked as failing.
ISTAT	12	State of the device, indicating if the device is ready or not.
RECRD	13	Maximum record size, set by INIT, INDAT or DATUT.
TERM	14	Character terminating an input record.
LAST	15	The last character which has been read.
BUFST	16	Start of buffer.
MAX	17	Buffer size (number of characters).
BHOLD	20	Available characters in the buffer.
HENTE	21	Pointer for fetching characters from the buffer.
CFREE	22	Free positions for characters in the buffer.
FYLLE	23	Pointer for putting characters into the buffer.

Name	Dis-placement	Meaning
		The following symbol is used for Teletype input only:
ALPHA	24	Circled alpha, character to call the operator communication.
		The following five symbols are used for operator communication device input:
LOGNH	25	Logical unit number of this device.
DW1	26	Working location for the operator communication
DW2	27	- " - - " -
DW3	30	- " - - " -
OPCOM	31	Nine locations holding the RT description for the operator communication.
	41	
		The following two symbols are used for output only:
ACT	14	RT program to be started, specified by DATUT.
ACTP1	15	Buffer pointer value when the RT program is to be started.
		The following two symbols are used for data lines only:
ENDCH	24	End code, marking end of block.
TERMF.	25	Terminator flag.
		The following five symbols are used for card readers only:
CSW	24	Card reader state.
BINP	25	Input pointer when the card is started.
SINP	26	Saved input pointer, used for removing trailing spaces.
CNUM	27	Column pointer.
SCNUM	30	Saved column pointer.

Contents of the entry DFLAG in the data field:

Name	Bit-position	Meaning
FAILB	0	Device failing; if set, the alternative unit will be used instead.
SKIPB	1	If this bit is set, the outputting program will never enter the waiting state; instead the output record will be skipped if the buffer is full.
HDUP	2	Bit marking a half duplex device.
ECHO	3	If this bit is set, the driver will produce an echo character.
CARD	4	This bit is set for card readers.
BCARD	5	Bit marking binary card reading.
	17	Mass storage device.

Mass Storage Data Field:

Name	Displace-ment	Meaning
TMSUB	-3	Address of routine used by TIMER.
TMR	-2	Timer counter.
TTMR	-1	Negative number of seconds before timeout.
DRIVR	0	Address of routine called by IDENT after mass storage interrupt.
STDEV	1	Routine used to start transfer.
MLINK	2	Link for monitor queue.
QUEUL	3	Queue element.
DFLAG	4	Status word. Bit 17 = 1 to identify the device as a mass storage device.
SAVT	5	Location used by IDENT.
SFLAG	6	Flag and device number word. Bit 0 : Always 0 Bit 7-0: Device number 0 - drum 1 - disc 1 2 - disc 2 3 - magnetic tape 1 4 - magnetic tape 2 5 - core/core 6 - plotter Bit 14 : Called by TIMER (Application level) Bit 15 : First exit from driver (Monitor level) Bit 16 : Device busy flag Bit 17 : Coreload transport
HDEV	7	Hardware device number (SNI device).
ACTON	10	Action program.
NXTRF	11	Pointer to RT head in stack. Zero if no waiting program.
TRNSF	12	Address of transfer routine (driver).
BUSY	13	Address of routine called after busy exit from driver.
FINSH	14	Address of routine called after finished exit from driver.

CONT.

Name	Displace-ment	Meaning
ERROR	15	Address of routine called after error exit from driver.
TRG	16	Register saved for next call of driver
ARG	17	
DRG	20	
XRG	21	
CTRG	22	Initial register content saved by STDEV
CARG	23	
CDRG	24	
CXRG	25	
BLSZ	26	Hardware block size.
HSTAT	27	Hardware status.
AERRB	30	Accumulated hardware status.
ERCNT	31	Error counter.
SERRB	32	Serious error bits.
WERRB	33	Write error message bits.
TACNS	34	Try again constant (negative).
TACOU	35	Try again counter.
WBACK	36	Write back error bits.
RADDR	37	Address of application level routine.
LOADN	40	Current coreload. (Only coreload device)
BACK	41	Write back flag. (Only coreload device)
SAVLD	42	Coreload wanted. (Only coreload device)
EOTST	37	EOT status. (Only magnetic tape.)

The last location in the coreload device data field is TFLAG. TFLAG is used to see if any devices transport to/from the coreload area. Each device corresponds to one bit in TFLAG. Thus the maximum numbers of mass storage devices are 16. The bit number in TFLAG corresponds to the device number saved in SFLAG in the data field.

Bit 0 - drum
" 1 - disc 1
" 2 - disc 2
" 3 - magnetic tape 1
" 4 - magnetic tape 2
" 5 - core/core
" 6 - plotter

TFLAG is only changed on monitor level. In all mass storage subroutines the B register must print to the data field for the actual device.

APPENDIX A

Operator Commands Summary

Operator Communication Commands		Further description on page	Short description
Command	Parameters		
ABORT	NAME	5-25	Terminate the RT-program.
ABSET	NAME, SEC, MIN, HOUR	5-24	Schedule the RT-program for execution when the given time is reached
ALT	LOG.UNIT1, LOG.UNIT2	5-22	Define the device corresponding to LOG.UNIT2 as alternative to LOG.UNIT1.
CLADJ	TIME, UNIT	5-25	Adjust internal calendar with the given number of time units.
COMP	CORELOAD	5-20	Compare a paper tape produced by PUNCH to the memory contents.
CONCT	NAME LINE	5-25	Connect the RT-program to the interrupt line.
CONT	NAME	5-22	Continue execution of the RT-program stopped after execution of a SINTRAN/ FORTRAN PAUSE statement.
D	NEWVALUE	5-19	Insert new contents in the location defined by the preceding EXAM command.
DATCL	....	5-21	Write current time and date on the Teletype.
DSCNT	NAME	5-25	Disconnect the RT-program from a time interval or/and an interrupt line.
DUMP	CORELOAD LOWLIMIT HIGHLIMIT	5-19	Print the defined memory block on the Teletype.
E	CORELOAD ADDRESS	5-19	Write the location contents on the Teletype.

CONT.

Operator Commands Summary CONT.

Operator Communication Commands		Further description on page	Short description
Command	Parameters		
FAIL	LOG.UNIT	5-22	The device defined is failing. Alternative unit, if any, should be used instead.
FTN		5-21	Start RT-FORTRAN compiler.
INTV	NAME, TIME, UNIT	5-25	Connect RT-program to time interval.
LOAD	....	5-21	Start on-line BRF-loader.
LOG	....	5-20	Write result of the performance logging on the Teletype.
LOGZ	NAME	5-20	Reset accumulated values of performance logging. The parameter indicates the RT-program (octal address to the RT-description) to be logged.
MAC	....	5-21	Start on-line Debug-MAC (MACD).
PIN	LOG.UNIT	5-21	Make a new I/O attempt on the device formerly timed out because of missing interrupts from the device.
PINC	LOG.UNIT	5-21	Clear device buffer and make a new I/O-attempt on the device formerly timed out.
PRIOR	NAME, PRIORITY	5-25	Change priority for the RT-program. If change is denied, the Operator Communication generates the error message NOPASS.
PUNCH	CORELOAD, LOWLIMIT HIGHLIMIT	5-19	Punch the defined memory block in a binary format.

CONT.

Operator Commands Summary CONT.

Operator Communication Commands		Further description on page:	Short description
Command	Parameters		
READ	CORELOAD	5-20	Deposit the paper tape (produced by PUNCH) contents in the core block defined by the coreload given in the command and the core addresses given on the binary tape.
RFAIL	LOG.UNIT	5-22	The device formerly defined as failing (FAIL) is now ready for use.
RTOFF	NAME	5-23	Inhibit start.
RTON	NAME	5-23	Allow start.
RSKIP	LOG.UNIT	5-22	Reset the skip-if-busy-mode.
RT	NAME	5-24	Schedule the RT-program for execution immediately.
RTP	MEM.PROT.REGISTER	5-23	Start program with special memory protect setting.
SET	NAME,TIME,UNIT	5-24	Schedule the RT-program for execution when given time is expired.
SKIP	LOG.UNIT	5-22	Put the output unit in a skip-if-full-mode.
STOP	....	5-21	Stop the computer (restart by pushing the CONTINUE-button).
UPDAT	MIN,HOUR,DAY,MONTH, YEAR	5-24	Update internal calendar to given contents (octal numbers).
WEQ		5-23	Write execution queue.
WRTS	NAME	5-24	Write static information in RT-description.
WRTT	NAME	5-24	Write temporary information.
WSTK		5-23	Write stack.
WTQ		5-23	Write time queue.

APPENDIX B

RT-LOADER COMMAND SUMMARY

Command Name	Parameter(s)	Description	Page No.
AC	-	Autoload core	4-2
AM	-	Autoload mass memory	4-2
CC	Core load number	Clear core load	4-3
CL	Common area label	Declare common label area in resident core	4-6
DE	Entry point name	Delete entry point	4-6
DP	RT program name	Delete RT program	4-7
EN	-	End load; include what was loaded	4-3
EQ	Entry point name, and address or defined name	Define entry point (Equality)	4-7
ER	Number of definitions	Equality read EQ s from input device	4-7
ES	Number of definitions	Equality read selectively EQ s from input device	4-8
EX	-	Exit the RT loader	4-4
FX	-	FIX, make linking table permanent	4-8
IN	Input device	Select input device	4-4
LP	-	Enable label printout (RT-FORTRAN)	4-5
MC	-	Manual load core	4-2
MM	-	Manual load mass memory	4-2
PD	Output device	Select output device for next W command	4-10
RF	Library symbol name	Make reference to a library symbol	4-5
RS	-	Reset the RT loader	4-5
RT	RT program name	Declare RT program name	4-9
WA	-	Write all symbols in linking table	4-10
WC	-	Write common area labels	4-10
WE	-	Write non-permanent entry points	4-10
WF	-	Write permanent (fixed) entry points	4-11
WG	-	Write global variable names	4-11
WL	-	Write RT program names	4-11
WP	-	Write real time programs	4-11
WR	-	Write referenced entry points	4-11

APPENDIX C

RUN TIME ERRORS

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
02	Illegal formal type (possible system error)	RT-prog.		Yes	RT-FORTRAN Run-time system
03	Wrong number of parameters	"		"	"
04	Disagreement between actual and formal parameter	"		"	"
06	Illegal actual type in subroutine call (possible system error)	"		"	RT-FORTRAN Run-time system I/O
07	Illegal block in parameter descriptor	"		"	RT-FORTRAN Run-time system
09	Parameter value on coreload by call of a subroutine on a different coreload	"		"	"
10	The subroutine has tried to modify a constant parameter	"		"	"
22	Wrong delimiter in READ/WRITE-statement (error in compiled code)	"		"	RT-FORTRAN Run-time system I/O
27	Wrong checksum on binary tape	"		"	Operator Communication
28	Illegal variable or format specification	"		"	ROUT
29	Parity error in ASCII input	"		"	"

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
35	Data line buffer full		Hardware device no.	No	SINTRAN I/O
36	Mass storage timeout		Unit nos. 0=drum 1=disc 1 2=disc 2 3=mag. tape 1 4=mag. tape 2 5=core/ core 6=plotter	"	"
37	I/O-device timeout		Log. unit no.	"	"
38	Card reader error		Column no.	"	"
39	Device error		Hardware device no.	"	"
48	Not loaded RT-program called for execution	RT-prog.		Yes	RT-Loader
49	Not loaded sub-program called	"		"	"
50	Transfer error Line 1: Line 2:	" Hardware state	 Unit no. see error 36 Block no.	"	Monitor
51	Parameter error in a call of TRANS	RT-prog.		"	"
52	Illegal time unit in SET, INTV, CLADJ or TIME	"		"	"
53	Stack overflow, or negative parameter in a call of STACK or PUSH	"		"	"

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal	Program aborted	Detected in
54	Negative or too great parameter in call of UNSTK. The call is ignored	RT-prog.		No	Monitor
55	Priority out of range in call of PRIOR. The call is ignored	"		"	"
56	Interrupt line number out of range in call of CONCT. The call is ignored	"		"	"
57	Wrong B-register contents in call of POP or PO	"		Yes	"
58	Parameter error in call of INHIB	"		"	"
59	Memory protect violation	"	P-register contents	"	"
60	Insufficient stack space before call of PUS	"		"	"
61	Coreload number too great	"		"	"
63	Parameter error in call of UPDAT	"		"	"
64	Parameter error in call of ABSET	"		"	"
65	Attempt to put an RT-program = 0 into time- or execution queue	"		No	"
66	Unidentified interrupt		Hardware interrupt level	"	SINTRAN I/O

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
71	Illegal character in format	RT-prog.		Yes	FORTRAN I/O
73	Attempt to fetch character beyond format	"		"	"
74	Attempt to store character beyond format	"		"	"
75	Illegal device number	"		"	"
76	Argument error, unidentified type specification	"		"	"
80	Trying to store characters beyond buffer	"		No	"
81	Trying to fetch characters beyond buffer	"		"	"
82	Buffer overflow on input	"		"	"
83	Wrong parity in input	"		"	"
84	Bad character in input	"		"	"
85	Integer overflow	"		"	"
86	Real overflow on input	"		"	"
87	Real overflow on input	"		"	"

CONT.

CONT.

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
88	Real overflow on output	RT-prog.		No	FORTRAN I/O
89	System error	"		Yes	"
90	Too great input record	"		"	"
91	I/O-error	"		"	"
99	Impossible I/O driver error	L-register	Hardware device no.	No	SINTRAN I/O

Error number	Meaning	xx (octal or name)	yy (decimal)	Program aborted	Detected in
AA	Error in <real> ** <real>	RT-prog.		No	FORTRAN Library
AI	Error in <real> ** <integer>	"		"	"
AT	Error in ATAN2	"		"	"
CH	Error in COSH	"		"	"
CO	Error in COS	"		"	"
DI	Error in integer division	"		"	"
EX	Error in EXP	"		"	"
IX	Error in <integer> ** <integer>	"		"	"
LN	Error in ALOG1, ALOG2 or ALOG	"		"	"
SH	Error in SINH	"		"	"
SI	Error in SIN	"		"	"
SQ	Error in SQRT	"		"	"
GO	Error in computed GOTO	"		"	FORTRAN Run-time System

A/S NORSK DATA-ELEKTRONIKK
Økernveien 145, Oslo 5 - Tlf. 21 73 71

COMMENT AND EVALUATION SHEET

Publ.No. ND-60.044.01
August 1973

SINTRAN II - Operator's Guide

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM _____

_____

_____